REMARKS

Claims 1-5, 8-11, and 14-18 are amended. Claims 19 and 20 are canceled without prejudice or disclaimer. No new matter is added by these amendments. Claims 1-18 are pending. By amending and canceling the claims, applicant is not conceding that the claims are unpatentable over the art cited by the Office Action and is not conceding that the claims are non-statutory under 35 U.S.C. 102, as the claim amendments and cancellations are only for the purpose of facilitating expeditious prosecution. Applicant respectfully reserves the right to pursue the subject matter of the claims as it existed prior to any amendment or cancellation in one or more continuation and/or divisional applications. Applicant respectfully requests reconsideration and allowance of all claims in view of the amendments above and the remarks that follow.

Rejections under 35 U.S.C. 102

Claims 1-20 are rejected under 35 U.S.C. 102(e) as unpatentable over Haydt (Pub. No. US 2004/0019882 A1). Applicant respectfully submits that the claims are patentable over Haydt because Haydt does not teach or suggest all elements of the claims for the reasons argued below.

Claim 1 recites: "receiving an allocate request from a queue pair," which is not taught or suggested by Haydt for the reasons argued below.

In contrast to claim 1, Haydt at [0010] describes that "one or more operations are inserted into one or more work queues." The Haydt "operations" do not teach or suggest the allocate request of claim 1, so Haydt does not teach or suggest "receiving an allocate request from a queue pair," as recited in claim 1.

Claim 1 further recites: "finding a free buffer associated with an entry in a free pool," which is not taught or suggested by Haydt for the reasons argued below.

Haydt at [0047] describes that "valid entries on a transmit queue include NoOp, send, RDMA write, RDMA read, and bind," which does not teach or suggest "finding a

free buffer in a free pool," as recited in claim 1 because NoOp, send, RDMA write, RDMA read, and bind are instructions or operations and not free buffers in a free pool.

Claim 1 further recites: "allocating the free buffer to the queue pair if the queue pair requests the free buffer for an operation having the operation type, the number of buffers allocatable to the queue pair is greater than zero, and the number of buffers allocated to the operation type is less than a maximum, wherein the allocating further comprises setting an identifier of the queue pair into the entry in the free pool and setting status in the entry in the free pool, wherein the setting the status in the entry in the free pool further comprises setting a use of the free buffer in the entry," which is not taught or suggested by Haydt for the reasons argued below.

In contrast to claim 1, Haydt at [0035] describes "a buffer key is used with a scatter/gather list entry to identify a virtual address range. A buffer can contain either application data or transport addresses. Each queue pair (i.e., work queues organized as send and receive queue pairs) contains a base local key index and base remote key index."

Thus, Haydt does not teach or suggest "allocating the free buffer to the queue pair if the queue pair requests the free buffer for an operation having the operation type, the number of buffers allocatable to the queue pair is greater than zero, and the number of buffers allocated to the operation type is less than a maximum, wherein the allocating further comprises setting an identifier of the queue pair into the entry in the free pool and setting status in the entry in the free pool, wherein the setting the status in the entry in the free pool further comprises setting a use of the free buffer in the entry," as recited in claim 1, because in Haydt the relationship of a queue pair to the base local key index and the base remote key index is unconditional and not conditional on a queue pair request, a number of buffers allocatable to the queue pair, a number of buffers allocated to an operation type, and a maximum, as recited in claim 1.

Claim 1 further recites: "receiving a validate request and a provided queue pair identifier from a direct memory access engine, wherein the validate request is associated

with a data transfer that uses the free buffer; in response to the receiving the validate request from the direct memory access engine, determining whether the provided queue pair identifier matches the queue pair identifier in the entry in the free pool, determining whether a requester of the data transfer matches the identifier of the queue pair in the entry in the free pool, and determining whether provided status provided by the requestor matches the status in the entry in the free pool; if the provided queue pair identifier matches the queue pair identifier in the entry in the free pool, the requester of the data transfer matches the identifier of the queue pair in the entry in the free pool, and the provided status provided by the requestor matches the status in the entry in the free pool, returning an indication of a successful validation; if the provided queue pair identifier does not match the queue pair identifier in the entry in the free pool, returning an error to the direct memory access engine; if the requester of the data transfer does not match the identifier of the queue pair in the entry in the free pool, returning the error to the direct memory access engine; and if the provided status provided by the requestor does not match the status in the entry in the free pool, returning the error to the direct memory access engine," which is not taught or suggested by Haydt for the reasons argued below.

In contrast to claim 1, Haydt at [0047] recites: "An RDMA read generates a completion queue entry when the data transfer completes."

The Haydt completion queue entry does not determine "whether the provided queue pair identifier matches the queue pair identifier in the entry in the free pool," as recited in claim 1. The Haydt completion queue entry does not determine "whether a requester of the data transfer matches the identifier of the queue pair in the entry in the free pool," as recited in claim 1. The Haydt completion queue entry does not determine "whether provided status provided by the requestor matches the status in the entry in the free pool," as recited in claim 1. The Haydt completion queue entry does not return "an indication of a successful validation" "if the provided queue pair identifier matches the queue pair identifier in the entry in the free pool, the requester of the data transfer matches the identifier of the queue pair in the entry in the free pool, and the provided status provided by the requestor matches the status in the entry in the free pool," as

recited in claim 1. The Haydt completion queue entry does not return "an error to the direct memory access engine" "if the provided queue pair identifier does not match the queue pair identifier in the entry in the free pool," as recited in claim 1. The Haydt completion queue entry does not return "the error to the direct memory access engine" "if the requester of the data transfer does not match the identifier of the queue pair in the entry in the free pool," as recited in claim 1. The Haydt completion queue entry does not return "the error to the direct memory access engine" "if the provided status provided by the requestor does not match the status in the entry in the free pool," as recited in claim 1.

Thus, Haydt does not teach or suggest all elements of claim 1. Claims 2-7 are dependent on claim 1, and are patentable over Haydt for the reasons argued above, plus the elements in the claims.

Claim 8 recites: "means for receiving an allocate request from a queue pair," which is not taught or suggested by Haydt for the reasons argued below.

In contrast to claim 8, Haydt at [0010] describes that "one or more operations are inserted into one or more work queues." The Haydt "operations" do not teach or suggest the allocate request of claim 8, so Haydt does not teach or suggest "means for receiving an allocate request from a queue pair," as recited in claim 8.

Claim 8 further recites: "means for finding a free buffer associated with an entry in a free pool," which is not taught or suggested by Haydt for the reasons argued below.

Haydt at [0047] describes that "valid entries on a transmit queue include NoOp, send, RDMA write, RDMA read, and bind," which does not teach or suggest "means for finding a free buffer in a free pool," as recited in claim 8 because NoOp, send, RDMA write, RDMA read, and bind are instructions or operations and not free buffers in a free pool.

Claim 8 further recites: "means for allocating the free buffer to the queue pair if the queue pair requests the free buffer for an operation having the operation type, the

number of buffers allocatable to the queue pair is greater than zero, and the number of buffers allocated to the operation type is less than a maximum, wherein the means for allocating further comprises setting an identifier of the queue pair into the entry in the free pool and setting status in the entry in the free pool, wherein the setting the status in the entry in the free pool further comprises setting a use of the free buffer in the entry," which is not taught or suggested by Haydt for the reasons argued below.

In contrast to claim 8, Haydt at [0035] describes "a buffer key is used with a scatter/gather list entry to identify a virtual address range. A buffer can contain either application data or transport addresses. Each queue pair (i.e., work queues organized as send and receive queue pairs) contains a base local key index and base remote key index."

Thus, Haydt does not teach or suggest "means for allocating the free buffer to the queue pair if the queue pair requests the free buffer for an operation having the operation type, the number of buffers allocatable to the queue pair is greater than zero, and the number of buffers allocated to the operation type is less than a maximum, wherein the means for allocating further comprises setting an identifier of the queue pair into the entry in the free pool and setting status in the entry in the free pool, wherein the setting the status in the entry in the free pool further comprises setting a use of the free buffer in the entry," as recited in claim 8, because in Haydt the relationship of a queue pair to the base local key index and the base remote key index is unconditional and not conditional on a queue pair request, a number of buffers allocatable to the queue pair, a number of buffers allocated to an operation type, and a maximum, as recited in claim 8.

Claim 8 further recites: "means for receiving a validate request and a provided queue pair identifier from a direct memory access engine, wherein the validate request is associated with a data transfer that uses the free buffer; means for determining whether the provided queue pair identifier matches the queue pair identifier in the entry in the free pool, determining whether a requester of the data transfer matches the identifier of the queue pair in the entry in the free pool, and determining whether provided status provided

by the requestor matches the status in the entry in the free pool, in response to the means for receiving the validate request from the direct memory access engine; means for returning an indication of a successful validation if the provided queue pair identifier matches the queue pair identifier in the entry in the free pool, the requester of the data transfer matches the identifier of the queue pair in the entry in the free pool, and the provided status provided by the requestor matches the status in the entry in the free pool; means for returning an error to the direct memory access engine if the provided queue pair identifier does not match the queue pair identifier in the entry in the free pool; means for returning the error to the direct memory access engine if the requester of the data transfer does not match the identifier of the queue pair in the entry in the free pool; and means for returning the error to the direct memory access engine if the provided status provided by the requestor does not match the status in the entry in the free pool," which is not taught or suggested by Haydt for the reasons argued below.

In contrast to claim 8, Haydt at [0047] recites: "An RDMA read generates a completion queue entry when the data transfer completes."

The Haydt completion queue entry does not determine "whether the provided queue pair identifier matches the queue pair identifier in the entry in the free pool," as recited in claim 8. The Haydt completion queue entry does not determine "whether a requester of the data transfer matches the identifier of the queue pair in the entry in the free pool," as recited in claim 8. The Haydt completion queue entry does not determine "whether provided status provided by the requestor matches the status in the entry in the free pool," as recited in claim 8. The Haydt completion queue entry does not return "an indication of a successful validation" "if the provided queue pair identifier matches the queue pair identifier in the entry in the free pool, the requester of the data transfer matches the identifier of the queue pair in the entry in the free pool, and the provided status provided by the requestor matches the status in the entry in the free pool," as recited in claim 8. The Haydt completion queue entry does not return "an error to the direct memory access engine" "if the provided queue pair identifier does not match the queue pair identifier in the entry in the free pool," as recited in claim 8. The Haydt

completion queue entry does not return "the error to the direct memory access engine" "if the requester of the data transfer does not match the identifier of the queue pair in the entry in the free pool," as recited in claim 8. The Haydt completion queue entry does not return "the error to the direct memory access engine" "if the provided status provided by the requestor does not match the status in the entry in the free pool," as recited in claim 8.

Thus, Haydt does not teach or suggest all elements of claim 8. Claims 9-14 are dependent on claim 8 and are patentable for the reasons argued above, plus the elements in the claims.

Claim 15 recites: "a controller that receives a plurality of allocate requests from a plurality of queue pairs," which is not taught or suggested by Haydt for the reasons argued below. In contrast to claim 15, Haydt at [0010] describes that "one or more operations are inserted into one or more work queues," so the Haydt "operations" do not teach or suggest the allocate request of claim 15.

Claim 15 further recites: "finds a plurality of free buffers in the free pool," which is not taught or suggested by Haydt for the reasons argued below. Haydt at [0047] describes that "valid entries on a transmit queue include NoOp, send, RDMA write, RDMA read, and bind," which does not teach or suggest "finds a plurality of free buffers in the free pool," as recited in claim 15 because NoOp, send, RDMA write, RDMA read, and bind are instructions or operations and not free buffers in a free pool.

Claim 15 further recites: "allocates the plurality of free buffers to the plurality of queue pairs if the plurality of queue pairs request the plurality of free buffers for an operation having the operation type, the number of buffers allocatable to the plurality of queue pairs is greater than zero, and the number of buffers allocated to the operation type is less than a maximum, sets an identifier of the plurality of queue pairs into the plurality of entries in the free pool, and sets status in the plurality of entries in the free pool, wherein the set of the status in the plurality of entries in the free pool further sets a use of the plurality of free buffers in the plurality of entries," which is not taught or suggested by Haydt for the reasons argued below.

In contrast to claim 15, Haydt at [0035] describes "a buffer key is used with a scatter/gather list entry to identify a virtual address range. A buffer can contain either application data or transport addresses. Each queue pair (i.e., work queues organized as send and receive queue pairs) contains a base local key index and base remote key index."

Thus, Haydt does not teach or suggest "allocates the plurality of free buffers to the plurality of queue pairs if the plurality of queue pairs request the plurality of free buffers for an operation having the operation type, the number of buffers allocatable to the plurality of queue pairs is greater than zero, and the number of buffers allocated to the operation type is less than a maximum, sets an identifier of the plurality of queue pairs into the plurality of entries in the free pool, and sets status in the plurality of entries in the free pool, wherein the set of the status in the plurality of entries in the free pool further sets a use of the plurality of free buffers in the plurality of entries," as recited in claim 15, because in Haydt the relationship of a queue pair to the base local key index and the base remote key index is unconditional and not conditional on a queue pair request, a number of buffers allocatable to the queue pair, a number of buffers allocated to an operation type, and a maximum, as recited in claim 15.

Claim 15 further recites: "receives a validate request and a provided queue pair identifier from a direct memory access engine, wherein the validate request is associated with a data transfer that uses one of the plurality of free buffers, determines whether the provided queue pair identifier matches the queue pair identifier in one of the plurality of entries in the free pool, determines whether a requester of the data transfer matches the identifier of one of the plurality of queue pairs in the one of the plurality of entries in the free pool, and determines whether provided status provided by the requestor matches the status in the one of the plurality of entries in the free pool, in response to the receive of the validate request from the direct memory access engine, returns an indication of a successful validation if the provided queue pair identifier matches the queue pair identifier in the one of the plurality of entries in the free pool, the requester of the data transfer matches the identifier of the one of the plurality of queue pairs in the one of the

plurality of entries in the free pool, and the provided status provided by the requestor matches the status in the one of the plurality of entries in the free pool, returns an error to the direct memory access engine if the provided queue pair identifier does not match the queue pair identifier in the one of the plurality of entries in the free pool, returns the error to the direct memory access engine if the requester of the data transfer does not match the identifier of the one of the plurality of queue pairs in the one of the plurality of entries in the free pool, and returns the error to the direct memory access engine if the provided status provided by the requestor does not match the status in the one of the plurality of entries in the free pool," which is not taught or suggested by Haydt for the reasons argued below.

In contrast to claim 15, Haydt at [0047] recites: "An RDMA read generates a completion queue entry when the data transfer completes."

The Haydt completion queue entry does not determine "whether the provided queue pair identifier matches the queue pair identifier in one of the plurality of entries in the free pool," as recited in claim 15. The Haydt completion queue entry does not determine "whether a requester of the data transfer matches the identifier of one of the plurality of queue pairs in the one of the plurality of entries in the free pool," as recited in claim 15. The Haydt completion queue entry does not determine "whether provided status provided by the requestor matches the status in the one of the plurality of entries in the free pool, in response to the receive of the validate request from the direct memory access engine," as recited in claim 15. The Haydt completion queue entry does not return "an indication of a successful validation if the provided queue pair identifier matches the queue pair identifier in the one of the plurality of entries in the free pool, the requester of the data transfer matches the identifier of the one of the plurality of queue pairs in the one of the plurality of entries in the free pool, and the provided status provided by the requestor matches the status in the one of the plurality of entries in the free pool," as recited in claim 15. The Haydt completion queue entry does not return "an error to the direct memory access engine if the provided queue pair identifier does not match the queue pair identifier in the one of the plurality of entries in the free pool," as recited in claim 15. The Haydt completion queue entry does not return "the error to the direct

memory access engine if the requester of the data transfer does not match the identifier of the one of the plurality of queue pairs in the one of the plurality of entries in the free pool," as recited in claim 15. The Haydt competition queue entry does not return "the error to the direct memory access engine if the provided status provided by the requestor does not match the status in the one of the plurality of entries in the free pool," as recited in claim 15.

Thus, Haydt does not teach or suggest all elements of claim 15. Claims 16-18 are dependent on claim 15 and are patentable for the reasons argued above, plus the elements in the claims.

Conclusion

Applicant respectfully submits that the claims are in condition for allowance and notification to that effect is requested. The Examiner is invited to telephone Applicant's attorney (651-645-7135) to facilitate prosecution of this application.

If necessary, please charge any additional fees or credit overpayment to Deposit Account No. 09-0465.

Respectfully submitted,

Date: March 11, 2008

Owen J. Gamon
Reg. No. 36,143
(651) 645-7135

IBM Corporation
Intellectual Property Law
Dept. 917, Bldg. 006-1
3605 Highway 52 North
Rochester, MN 55901

CERTIFICATE UNDER 37 C.F.R. 1.8

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to Mail Stop AF, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, or is being transmitted via facsimile to the U.S. Patent and Trademark Office, 571-273-8300, on: March 11, 2008.

Owen J. Gamon
Registration No. 36,143